OSS に対する ANP と SRGM に基づく最適バージョンアップ時期の推定問題に関する一考察(不確実性の下での意思決定と数理モデル)

AUTHOR(S):

田村, 慶信; 山田, 茂

CITATION:

田村, 慶信 ...[et al]. OSS に対する ANP と SRGM に基づく最適バージョンアップ時期の推定問題に関する一考察(不確実性の下での意思決定と数理モデル). 数理解析研究所講究録 2006, 1477: 199-207

ISSUE DATE:

2006-03

URL:

http://hdl.handle.net/2433/48253

数理解析研究所講究録 1477 巻 2006 年 199-207

OSSに対するANPとSRGMに基づく最適バージョンアップ時期の推定問題に関する一考察

鳥取環境大学・環境情報学部
情報システム学科
田村 慶信 (Yoshinobu Tamura)
Department of Information Systems,
Faculty of Environmental and Information Studies,
Tottori University of Environmental Studies
E-mail: tamura@kankyo-u.ac.jp

鳥取大学・工学部
社会開発システム工学科
山田 茂 (Shigeru Yamada)
Department of Social Systems Engineering,
Faculty of Engineering,
Tottori University
E-mail: yamada@sse.tottori-u.ac.jp

概 要

最近のソフトウェア開発は，クライアント/サーバ・システムやWebプログラミング，オブジェクト指向開発，ネットワーク環境での分散開発といった新しい開発形態が多用されるようになってきている．特に，世界中に分散した無数のモジュールいわゆるコンポーネントに対して，多くの開発者が協調しながら開発を行うという特徴をもつオープンソースプロジェクトは，近年特に注目されている．

本論文では，こうしたオープンソースプロジェクトの下で開発されたオープンソースソフトウェアに対して，意思決定手法の1つであるANP（Analytic Network Process）手法を用いて各コンポーネントに対する重要度を推定する．同時に，ソフトウェア信頼度成長モデルに基づき各コンポーネント間の相互作用を包括した信頼性評価法を提案する．さらに，最適なバージョンアップ時期の決定法について考察する．

1 はじめに

最近のソフトウェア開発は，クライアント/サーバ・システムやWebプログラミング，オブジェクト指向開発，ネットワーク環境での分散開発といった新しい開発形態が多用されるようになってきている．現在，分散ソフトウェア共同開発は，同一企業内における開発形態から，複数のソフトウェアハウスや同一企業内，複数の企業間での遠隔地間共同開発，さらには，多くの開発者が協調しながら開発を行うオープソースプロジェクトなどの様々な形態が存在する [1].

特に，オープンソースプロジェクトは，世界中に分散した無数のモジュールいわゆるコンポーネントが，何らかのプラットフォーム上で的確に機能しているとすれば，開発が迅速であるばかりか，競争原理によりある評価基準の下でベストなものが残っていくと考えられる．オープンソースプロジェクトが分散型開発モデルを採用して成功した例として，GNU/Linuxオペレーティングシステム[1]やApacheウェブサーバなど[2]が挙げられる．

特に，オープンソースソフトウェア（open source software，以下OSSと略す）は，ユーザの使用により不具合が確認されるとバグトラッキングシステム上に不具合内容が報告され，その内容に基づきソースコードの修正作業を開発者が行い，修正されたOSSを再度，公表・配布するという開発サイクルで成り立っている．このように，OSSでは開発から運用保守におよぶ工程においてソフトウェアの信頼性を評価するという試みが行われていなかった．オープンソースプロジェクトのメンバー構成と動機付けの仕組みを考えた場合，中心にコアがあり，それを複数の周辺レベルが互いに混ざり合って取り囲む構造になっている．特に，開発ボランティアは，コア開発者と周辺開発者に分類される．最重要のメンバーは中心的なソフトウェアを担当する開発者であり，彼らは中心的なメーリングリストにアクセス可能となっている．この指導的集団を形成している主要メンバーが主導的にテスト進捗度管理技術を導入することによって，より高品質なOSSの開発に結びつくものと考える．特に，一般企業において実践されているテスト進捗度管理技術をOSSの開発サイクルに組み込むことによって，従来よりもより高品質なOSSの開発が可能となると思われる．

本論文では，こうしたオープンソースプロジェクトの下で開発が進められているソフトウェアとして，メウィンドウシステム用のXfce[2]と呼ばれるデスクトップ環境を取り上げる．従来から，ソフトウェア製品の開発プロセ

[1]Linuxは，Linus Torvaldsの米国およびその他の国における登録商標あるいは商標です．
[2]その他記載している会社名，商品名は一般に各社の商標または登録商標です．

スにおけるテスト進捗管理や出荷品質の把握のための信頼性評価を行うアプローチとして，ソフトウェア故障の発生現象を不確定事象として捉えて確率・統計論的に取り扱う方法がとられている．その1つが，ソフトウェア信頼度成長モデル（software reliability growth model，以下SRGMと略す）である[3]．これまでに，分散ソフトウェア開発環境を対象としたSRGMがいくつか提案されているが，アーキテクチャ，マシン構成などの組み合わせに自由度をもつことから，分散開発環境における有効なテスト方法は提案されていないのが現状である．

分散共同開発環境におけるソフトウェアの信頼性評価手法の開発において，各コンポーネントでのデバッグの状況やその良し悪しが，システム全体の信頼性に与える影響を考慮しようとする場合，プログラムパス，コンポーネントの規模，フォールト報告者のスキルなどの，様々に絡み合った要因を捉える必要があると考えられる．こうした分散共同開発環境において，各コンポーネントがシステム全体の信頼性に与える影響を考慮するために，本論文では，意思決定手法の1つであるANP（Analytic Network Process）手法を用いて各コンポーネントに対する重要度を推定する．同時に，SRGMに基づき各コンポーネント間の相互作用を包括した信頼性評価法を提案する．また，実際のフォールト発見数データに対する数値例を示す．さらに，OSSの最適なバージョンアップ時期の決定法についても考察する．

2 各コンポーネントに対する信頼性評価

2.1 SRGMに基づく信頼性評価

従来から，ソフトウェアの信頼性を定量的に評価する手法として，SRGMによる方法がとられている．中でも非同次ポアソン過程（nonhomogeneous Poisson process，以下NHPPと略す）モデルは，実利用上極めて有効でありモデルの簡潔性が高いゆえにその適用性も高く，実際のソフトウェア信頼性評価に広く応用されている．このNHPPモデルは，所定の時間区間内に発見されるフォールト数や発生するソフトウェア故障数を観測して，これらの個数を数え上げる計数過程$\{N(t), t \geq 0\}$を導入し，以下の式で与えられる確率変数すなわちポアソン過程を仮定するSRGMである[3]：

$$\Pr\{N(t)=n\} = \frac{\{H(t)\}^n}{n!}\exp[-H(t)] \quad (n=0,1,2,\cdots). \tag{1}$$

ここで，$\Pr\{\cdot\}$は確率を表し，$H(t)$は時間区間$(0,t]$において発見される総期待フォールト数，すなわち$N(t)$の期待値を表し，NHPPの平均値関数と呼ばれる．

本論文では，各コンポーネントについて累積フォールト発見数データの成長曲線の形状により，以下に示すNHPPモデル[3]のうち最適なモデルを適用する．

- 指数形SRGM
- 習熟S字形SRGM

さらに，モデルに含まれる未知パラメータの推定方法として最尤法を適用する．

2.2 指数形SRGM

残存フォールト1個当りのフォールト発見率，または残存フォールト1個当りのソフトウェア故障発生率は一様であり，指数形SRGMは$b(t)$が時刻tに関して一定であるので，一定型フォールト発見率をもつと言われる．その平均値関数$E_i(t)$は，次式によって与えられる．

$$E_i(t) = a_i(1-e^{-b_i t}) \quad (a_i>0,\ b_i>0). \tag{2}$$

ここで，$E_i(t)$は$i(i=1,2,\cdots,n)$番目のソフトウェアコンポーネントに対して適用された指数形SRGMの平均値関数であり，時間区間$(0,t]$において発見される累積フォールト数の期待値を表す．パラメータa_iは最終的に発見される総期待フォールト数，パラメータb_iはフォールト1個当りのソフトウェア故障発見率またはフォールト発見率を表す．

2.3 習熟S字形SRGM

習熟S字形SRGMの平均値関数は，

$$I_i(t) = \frac{a_i(1-e^{-b_i t})}{(1+c_i \cdot e^{-b_i t})} \quad (a_i > 0,\ b_i > 0,\ c_i > 0), \tag{3}$$

により与えられる．ここで，$I_i(t)$ は i 番目のソフトウェアコンポーネントに対して適用された習熟S字形SRGMの平均値関数であり，時間区間 $(0,t]$ において発見される累積フォールト数の期待値を表す．また，a_i は，各コンポーネントにおいて最終的に発見される総期待フォールト数を表す定数パラメータである．さらに，パラメータ b_i は各コンポーネントにおけるフォールト1個当りの発見率，および c_i はテストに対する習熟性を表す習熟係数を表す．

2.4 適用モデル選択のための適合性評価基準

本論文では，各コンポーネントに対して適用される **2.2** および **2.3** の各SRGMについて，赤池情報量規準（Akaike's information criterion，以下AICと略す）および平均偏差2乗和（mean squared errors，以下MSEと略す）を評価基準としたモデルの選択を行う．

AICは，観測データがモデルにどの程度適合するかを判定する規準であり，モデルのもつパラメータ数も考慮して，値が最も小さくなるモデルが最もよいと判断する．ここで，AICは数値の大小ではなく，数値の差の大小に意味があり，それは1~2程度以上の差であれば2つのモデルの適合性には有意な差があるとし，1以下のとき有意ではないと判断する．一般に，適当なモデルMを仮定すると，次式によって与えられる：

$$\mathrm{AIC(M)} = -2 \cdot \mathrm{MLL(M)} + 2 \cdot P. \tag{4}$$

ここで，MLL(M)はモデルMの最大対数尤度，P はモデルMに含まれる独立な未知パラメータの数である．

また，MSEは実測値と推定値との2乗誤差をデータ数で平均化したものである．ここで一定のテスト時刻 t_k までに発生した累積フォールト数 y_k に関する K 組の累積発見フォールトデータ $(t_k, y_k)(k = 1, 2, \cdots, K)$ が観測されているものとすると，

$$\mathrm{MSE} = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2, \tag{5}$$

により与えられる．ここで，$\hat{y}_k$ はテスト時刻が $t_k(k = 1, 2, \cdots, K)$ のときの推定値を表す．

AICはモデルのもつパラメータ数を考慮した適合性評価尺度として知られているが，その値の差が1以下である場合には有意でないと判断される．したがって，本論文では，より確実に実測データに対する適合性を判断するために，上記の2つの評価基準に基づきモデルを選択する．具体的には，AICを1番目の評価基準とし，AICの値の差が1以上であれば有意であるものと判断する．また，AICの値の差が1以下であれば有意でないものと判断し，その場合には2番目の評価基準であるMSEに基づいたモデルの選択を行う．

2.5 ANPに基づく重み係数の推定

OSSに対するソフトウェア信頼性評価手法の開発において，各コンポーネントでのデバッグの状況やその良し悪しが，システム全体の信頼性に与える影響を考慮しようとする場合，プログラムパス，コンポーネントの規模，フォールト報告者のスキルなどの，様々に絡み合った要因を捉える必要があると考えられる．しかしながら，これらを考慮することは困難であることが予想される．これまでに，こうした複雑な状況下でシステム全体の信頼性に対する各コンポーネントの影響度合いを推定するために，主観的判断の合理的合成方法として知られているAHP[4]を利用し，システム全体の信頼性に対する各コンポーネントの重要度を表す重み係数を推定する方法が提案されている[5]．1970年代に開発されたAHPは，主観的判断による意思決定支援に有効な方法として，欧米を中心に経営問題，エネルギー問題，政策決定，都市計画学など様々な分野で広く活用されている．

本論文では，評価基準と代替案との間の依存関係を考慮したネットワーク型分析法であるANPに基づきシステム全体の信頼性に対する各コンポーネントの重要度を推定する[6]．ANPはAHPの階層構造をネットワークモデルに拡張したものであり，評価基準と代替案との間の依存関係を組み込んで，評価することが可能となる．

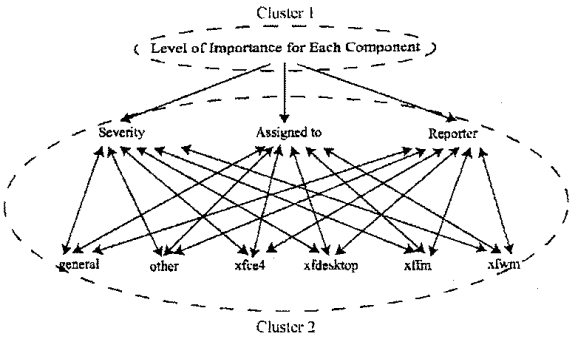

図 1： 本論文における ANP のネットワーク構造.

まず，実測データに対する適用事例を示すために，UNIX 系 OS 上で動作するデスクトップ環境の Xfce と呼ばれる OSS を 1 例にとる．そのネットワーク構造を図 1 に示す．ここで，フォールトの重要度（Severity），フォールト修正者のスキル（Assigned to），フォールト報告者のスキル（Reporter）の 3 つを評価基準とし，代替案としては，general，other，xfce4，xfdesktop，xffm，xfwm の 6 つのコンポーネントを割り当てた．このとき，超行列は，

$$S = \begin{bmatrix} A_1 & 0 \\ B_{21} & A_2 \end{bmatrix}, \tag{6}$$

と表され，$A_1 = 0$，$B_{21} = \begin{bmatrix} v \\ 0 \end{bmatrix}$，$A_2 = \begin{bmatrix} 0 & W \\ U & 0 \end{bmatrix}$，となる．ここで，$A_i$ はクラスタ i の中での評価行列，B_{ji} はクラスタ i からクラスタ j による評価行列を表す．また，v は各評価基準の重要度を表し，U は評価基準が代替案を，W は代替案が評価基準を評価する評価行列を表す．この v，U，W は，Xfce の Web 上で公開されているバグトラッキングシステムから採取されたデータに基づき，各評価基準ごとの各コンポーネントのウエイトを一対比較することにより求められる．フォールト重要度に対する各コンポーネントの重み付けは致命的なフォールトの数を，フォールト修正者に対する各コンポーネントの重み付けは，システム全体に精通し，すべてのコンポーネントのフォールト修正に最も影響を与える者を，およびフォールト報告者に対する各コンポーネントの重み付けは，システム全体に精通し，全てのコンポーネントに対してフォールトを最も報告している者を基準とした．各評価基準に対する重み付けに関しては，本来ならば Xfce の開発に携わるコア開発者により行われることが理想的であるが，本論文では著者の主観により行っている．

まず，行列を基準化するために対角ブロックにある部分行列 $A_1, A_2, \cdots, A_n$ の最大固有値 $\lambda_1, \lambda_2, \cdots, \lambda_n$ を求め，以下の計算を行う．

$$\overline{A_i} = \frac{1}{\lambda_i} A_i \quad (i = 1, 2, \cdots, n), \tag{7}$$

$$\overline{B_{ki}} = \frac{1}{\lambda_i} B_{ki} \quad (k = i+1, i+2, \cdots, n). \tag{8}$$

ここで，もし A_i がスカラーで 0 のとき，$\lambda_i = 1$ とすると，

$$S = \begin{bmatrix} \overline{A_1} & 0 \\ \overline{B_{21}} & \overline{A_2} \end{bmatrix}, \tag{9}$$

となる．次に，式 (9) で表される超行列の第 2 番目の行である，$\begin{bmatrix} \overline{B_{21}} & \overline{A_2} \end{bmatrix}$ を抽出し，この行列の第 i 行の正の要素の個数を n_{2i} とし，この第 i 行の各要素を n_{2i} で割った行列を $\begin{bmatrix} \widehat{B_{21}} & \widehat{A_2} \end{bmatrix}$ とする．さらに，

$$\widehat{b_2} = \widehat{B_{21}} u_1, \tag{10}$$

を求める．ここで，クラスタ1が唯一個の成分からなる場合 $u_i = 1$ と仮定する．この $\widehat{b_2}$ はクラスタ1からクラスタ2へ与えられる評価値である．

以上のことから，

$$\widehat{b_2} + \widehat{A_2} u_2 = u_2, \tag{11}$$

を満たす u_2 を求めることで，システム全体の信頼性に対する各コンポーネントの重要度を表す重み係数 $p_i (i = 1, 2, \cdots, n)$ を推定することができる [7, 8]．

3 システム全体に対する信頼性評価

本論文では，システム全体の信頼性を評価するために，各コンポーネントに対して適用されたSRGMに含まれる未知パラメータを最尤法により推定された結果を踏まえて，以下に示すSRGMを適用する．

3.1 対数型ポアソン実行時間モデル

本論文で取り上げるOSSの動作環境は，様々なアプリケーションソフトウェアから影響を受け易く，従来のような同一組織内で開発され，単体で動作するソフトウェアシステムとは大きく環境が異なる．こうしたソフトウェア間の相互作用により，発見されるフォールト数も一定の値に収束することなく，将来的には増加し続けるものと考えられる．

本論文では，検出可能フォールト数が無限であると仮定されたNHPPに基づく対数型ポアソン実行時間モデルを適用する．時間区間 $(0, t]$ で発見される総期待フォールト数を表す平均値関数 $\mu(t)$ は，

$$\begin{aligned} \mu(t) &= \frac{1}{\theta - P} \ln[\lambda_0(\theta - P)t + 1] \\ &\quad (0 < \theta,\ 0 < \lambda_0,\ 0 < P < 1), \end{aligned} \tag{12}$$

により与えられる．ここで，パラメータ λ_0 は初期故障強度，パラメータ θ はソフトウェア故障1個当りの故障強度の減少率を表す．また，パラメータ P はシステム全体に及ぼすコンポーネントの影響率を表す．これは，各コンポーネントに対して推定されたパラメータ b_i と **2.5** のANP手法により推定された重みパラメータ p_i との重み付き平均により表されるものとし，

$$P = \frac{\sum_{i=1}^{n} p_i \cdot b_i}{\sum_{i=1}^{n} p_i} = \sum_{i=1}^{n} p_i \cdot b_i, \tag{13}$$

により定義する．ここで，n はソフトウェアシステムのコンポーネント数を表す．さらに，p_i は各コンポーネントに対する重みパラメータを表し，システム全体に対する各コンポーネントの重要度を表す．また，b_i は i 番目のコンポーネントに対する指数形SRGMおよび習熟S字形SRGMに含まれるフォールト発見率を表す定数パラメータを表す [5, 9]．

3.2 ソフトウェア信頼性評価尺度

式 (12) の平均値関数をもつNHPPモデルから，種々のソフトウェア信頼性評価のための定量的尺度を導出できる．

時刻 t までシステムの運用が行われているときに，時間区間 $(t, t + x]$ $(t \geq 0, x \geq 0)$ においてソフトウェア故障の発生しない条件付き確率は，

$$R(x|t) = \exp[\mu(t) - \mu(t + x)], \tag{14}$$

となり，ソフトウェア信頼度（software reliability）と呼ばれる．

平均ソフトウェア故障発生時間間隔（mean time between software failures: MTBF）は，ソフトウェア 故障の発生頻度を表すのに有益な尺度である．また，MTBFが大きな値を取ることは，それだけフォールトが発見し難くなり，ソフトウェア信頼性が向上したと判断できることになる．任意の時刻 t における瞬間MTBF（instantaneous MTBF: $\mathrm{MTBF_I}$）および累積MTBF（cumulative MTBF: $\mathrm{MTBF_C}$）は，以下のように導出できる．

表 1： ANP に基づく各コンポーネントに対する重み係数の推定結果.

Component	Weight parameter p_i
general	0.055030
other	0.44428
xfce4	0.093274
xfdesktop	0.18277
xffm	0.12294
xfwm	0.10170

任意の時刻 t における瞬間的なフォールト発見間隔の平均を意味する瞬間 MTBF は，

$$MTBF_I(t) = \frac{1}{d\mu(t)/dt}, \tag{15}$$

となる.

運用開始時点から考えたときの発見フォールト 1 個当りに要する発見時間の平均を意味する累積 MTBF は，

$$MTBF_C(t) = \frac{t}{\mu(t)}, \tag{16}$$

により表すことができる.

4 実測データに適用した数値例

4.1 各コンポーネントに対する信頼性評価

Xfce[2] のフォールトデータを適用した数値例を示す．本論文で用いたデータは，複数のコンポーネントから構成された Xfce デスクトップ環境におけるバグトラッキングシステムから採取されたものである.

各コンポーネントの累積フォールト発見数データを図 2 に示す．次に，**2.5** の ANP に基づく各コンポーネントに対する重みパラメータ $p_i(i = 1, 2, \cdots, n)$ の推定結果を表 1 に示す．ここで，各評価基準に対する重み付けに関しては，本来ならば Xfce の開発に主導的立場にあるコアメンバーにより行われることが理想的であるが，本論文では著者の主観により行っている．特に，本論文で取り上げる評価基準は簡単のために 3 つとする．すなわち，Xfce のバグトラッキングシステムから利用可能なデータとしてシステム全体に最も影響を及ぼしていると考えれる，各コンポーネントに対するフォールトの重要度（Severity），フォールト修正者（Assigned to），およびフォールト報告者（Reporter）を取り上げた．表 1 から，other コンポーネントに対する重要度が最も大きいことが分かる．一方，general コンポーネントに対する重要度は最小であることが確認できる．特に，other コンポーネントについては 2004 年 9 月末から開発が開始されている新しいコンポーネントであることから，システム全体に対する重要度が極端に高くなっていることが考えられる.

4.2 システム全体に対する信頼性評価

次に，各コンポーネントに対して適用された SRGM に含まれる未知パラメータを最尤法により推定された結果を踏まえて，Xfce デスクトップ環境のソフトウェア信頼性評価の一例を示す．式 (12) における累積フォールト発見数の期待値の推定値 $\widehat{\mu}(t)$ を図 3 に示す．さらに，式 (14) における推定されたソフトウェア信頼度 $\widehat{R}(t)$ を図 4 に示す．図 5 は，式 (15) における推定された瞬間 MTBF $\widehat{MTBF_I}(t)$ を表す．また，式 (16) における推定された累積 MTBF $\widehat{MTBF_C}(t)$ を図 6 に示す.

5 最適バージョンアップ

5.1 最適バージョンアップ時期の推定

OSS の開発において，ある程度目安となるような適切なバージョンアップ時期を推定することは，リリース後の信頼性維持や進捗度管理に役立つと考えられる．本論文では，オープンソースプロジェクトの下で開発された Xfce を一例に挙げ，OSS のバージョンアップ時期の推定方法について議論する.

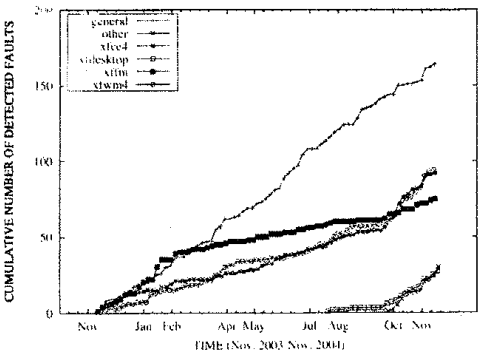

図 2： 各コンポーネントの累積フォールト発見数データ.

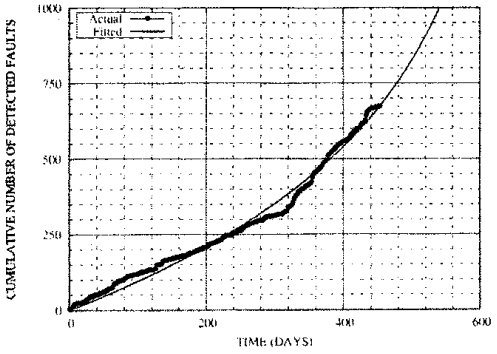

図 3： 推定された累積フォールト発見数の期待値, $\widehat{\mu}(t)$.

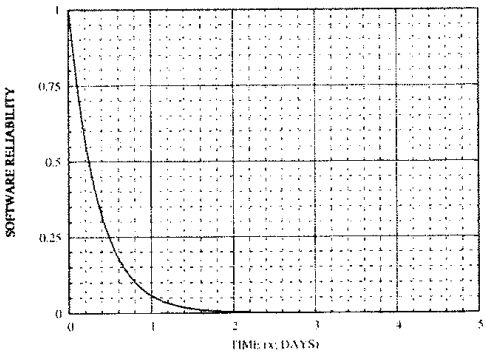

図 4： 推定されたソフトウェア信頼度 $\widehat{R}(t)$.

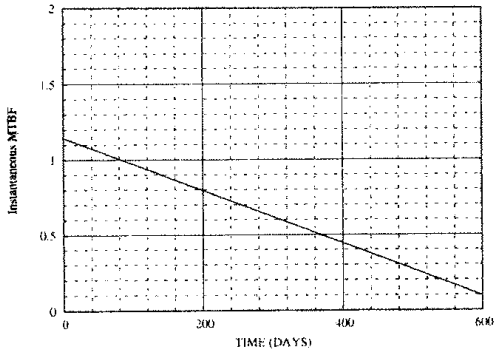

図 5： 推定された $\widehat{MTBF_I}(t)$.

バージョンアップには大きく分けてマイナーバージョンアップとメジャーバージョンアップがある．しかしながら，どの程度の改訂で区別されるという明確な基準がある訳ではない．マイナーバージョンアップは，旧版にいくつかの細かい機能が追加されたり，性能が若干向上した場合に実施される．ただし，機能の不具合やセキュリティ上の脆弱性を修復するような，クリティカルなフォールトがいくつか発見され緊急に修正を施す必要がある場合に実施される改訂はマイナーバージョンアップではなく，バグフィックスと呼ばれている．一方，メジャーバージョンアップは，OSS 自体の機能が大きく変更されたり，大型の新機能の追加や，性能が劇的に向上した場合に実施される．OSS におけるマイナーバージョンアップは，不定期かつ頻繁に実施されていることから，その推定時期を予測することは困難であり，たとえマイナーバージョンアップ時期が推定できたとしても，その有益性は小さいと考えられる．したがって，本論文では，メジャーバージョンアップを対象とし，OSS がベータ版として公開されている段階から，実際に正式リリースされる時期を予測するような手法について考察する．

5.2 総開発労力の定式化

OSS の開発に伴う総労力を定式化し，総開発労力を最小にする時刻を最適バージョンアップ時刻と定義することにより，バージョンアップ後の信頼性維持や進捗度管理に役立つものと考える．まず，総開発労力を定式化するために，以下のパラメータを定義する．

$m_{1,i}$: コンポーネント i のフォールト修正に伴う修正労力 $(m_{1,i} > 0)$,

$m_{2,i}$: コンポーネント i の単位時間当りの開発労力 $(m_{2,i} > 0)$,

m_3: メジャーバージョンアップ後のフォールト修正に伴う保守労力 $(m_3 > 0)$,

m_4: メジャーバージョンアップ後の単位時間当りの保守労力 $(m_4 > 0)$.

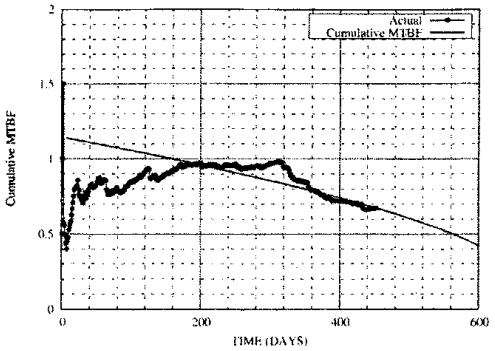

図 6： 推定された $\widehat{MTBF}_C(t)$.

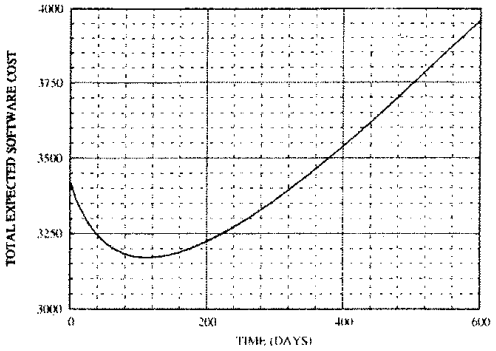

図 7： 推定された総期待開発労力.

よって，以下のような各コンポーネントに対する期待開発労力が得られる．

$$W(t_i) = \sum_{i=1}^{n} \{m_{1,i} H_i(t_i) + m_{2,i} t_i\} \qquad (i = 1, 2, \cdots, n). \tag{17}$$

ここで，$H_i(t_i)$ は式 (2) と式 (3) における指数形 SRGM または習熟 S 字形 SRGM の平均値関数を表し，t_i はコンポーネント i の運用時間を表す．

一方，メジャーバージョンアップ後の保守労力は以下のように定式化できる．

$$R(t) = m_3\{\mu(t_0) - \mu(t)\} + m_4 t. \tag{18}$$

ここで，t_0 は目標 MTBF の到達時点を表す．

したがって，総期待開発労力は，式 (17) および式 (18) より，

$$\begin{aligned} WR(t) &= W(t_i) + R(t) \\ &= \sum_{i=1}^{n} \{m_{1,i} H_i(t_i) + m_{2,i} t_i\} + m_3\{\mu(t_0) - \mu(t)\} + m_4 t, \end{aligned} \tag{19}$$

のように表すことができる．この式 (19) を最小にする時刻 t^* が，OSS の最適メジャーバージョンアップ時刻となる．

5.3 数値例

ここで取り上げる Xfce は，開発後間もない成長過程にある OSS であり，2005 年 7 月時点におけるメジャーバージョンアップの更新回数は 1 回のみであることから，過去の経験に基づいて目標累積 MTBF の値を決定することが不可能である．したがって，ここでは一例として目標累積 MTBF の値を 1 日と仮定する．式 (19) における推定された総期待開発労力を図 7 に示す．図 7 から，目標 MTBF を 1.0 と仮定した場合における最適メジャーバージョンアップ時期は 113 日目となり，そのときの総期待開発労力は 3127.6 であることが確認できる．

6 おわりに

本論文では，オープンソースプロジェクトの下で開発された OSS に対するソフトウェア信頼性評価法について議論した．特に，ANP 手法と既存の SRGM を融合することにより，各コンポーネント間の相互作用を包括した信頼性評価法を提案した．また，実際の Linux デスクトップ環境の 1 つである Xfce に基づいたフォールト発見数データに対する数値例を示すとともに，本手法の適用可能性について議論した．

これまで，OSS の開発工程ではソフトウェアの信頼性を定量的に評価するという試みが行われていなかった．したがって，本論文において新たに提案されたソフトウェア信頼性評価技術をオープンソースプロジェクトに導入することによって，開発者の主観を取り込んだ形で，より高品質な OSS の開発に結びつくものと考える．さらに，OSS の開発において，ある程度目安となるような適切なバージョンアップ時期を推定するために，最適バージョ

ンアップ時期の推定方法について議論した．本論文において提案された手法によって，OSSのメジャーバージョンアップ後の信頼性維持や進捗度管理に役立つと考えられる．また，最適バージョンアップ時期の推定のために，OSSの総期待開発労力を定式化した．しかしながら，フォールト修正に伴う修正労力や単位時間当りの開発労力に関するパラメータの決定方法が曖昧であることから，今後はこれらのパラメータの厳密な設定方法について議論する必要がある．

こうしたオープンソースプロジェクトに基づく分散共同開発形態は，今後も急速に発展するものと思われる．特に，企業システムでのOSSの活用や，異なる企業間における分散開発形態の1手段として，同様な開発環境が利用されていることから，こうした分散共同開発環境において，システム全体に与える各コンポーネントの重要度を包括した信頼性評価法として利用できるものと考える．

謝辞

本論文の一部は，文部科学省科学研究費基盤研究(C)(2)（課題番号15510129）および若手研究(B)（課題番号17700039）の援助を受けたことを付記する．

参考文献

[1] A. Umar, *Distributed Computing and Client-Server Systems*, Prentice Hall, New Jersey, 1993.

[2] Olivier FOURDAN, Xfce – Desktop Environment, http://www.xfce.org/

[3] 山田 茂, ソフトウェア信頼性モデル－基礎と応用－, 日科技連出版社, 東京, 1994.

[4] T. Satty, *The Analytic Hierarchy Process*, McGraw-Hill, New York, 1980.

[5] 田村慶信, 山田茂, 木村光宏, "オープンソース共同開発環境に対するソフトウェア信頼性評価法に関する考察," 電子情報通信学会論文誌, Vol. J88-A, No. 7, pp. 840–847, 2005.

[6] S. Iriguchi, Y. Tamura and S. Yamada, "A reliability assessment method based on ANP for an open source software," *Proceedings of the 11th ISSAT International Conference on Reliability and Quality in Design*, St. Louis, Missouri, USA, August 4–6, 2005, pp. 12–16.

[7] 木下栄蔵, AHPの理論と実際, 日科技連出版社, 東京, 2000.

[8] 木下栄蔵, 入門AHP 決断と合意形成テクニック, 日科技連出版社, 東京, 2000.

[9] Y. Tamura and S. Yamada, "Comparison of software reliability assessment methods for open source software," *Proceedings of the 11th IEEE International Conference on Parallel and Distributed Systems (ICPADS2005)–Volume II*, Fukuoka, Japan, July 20–22, 2005, pp. 488–492.